Accelerating Benders method using covering cut bundle generation

Georgios K. D. Saharidis, Michel Minoux and Marianthi G. Ierapetritou

Department of Chemical and Biochemical Engineering, Rutgers – The State University of New Jersey, 98 Brett Road, Piscataway, NJ 08854-8058, USA,
Laboratoire d’Informatique de Paris 6, Université Pierre, Marie Curie, Place Jussieu, 75005 Paris, France
E-mail: saharidis@gmail.com

Received 16 May 2008; received in revised form 25 December 2008; accepted 7 February 2009

Abstract

Over the years, various techniques have been proposed to speed up the classical Benders decomposition algorithm. The work presented in the literature has focused mainly on either reducing the number of iterations of the algorithm or on restricting the solution space of the decomposed problems. In this article, a new strategy for Benders algorithm is proposed and applied to two case studies in order to evaluate its efficiency. This strategy, referred to as covering cut bundle (CCB) generation, is shown to implement in a novel way the multiple constraints generation idea. The CCB generation is applied to mixed integer problems arising from two types of applications: the scheduling of crude oil and the scheduling problem for multi-product, multi-purpose batch plants. In both cases, CCB significantly decreases the number of iterations of the Benders method, leading to improved resolution times.

Keywords: Benders decomposition; covering cut bundle; mixed integer programming; multi-generation of cuts

1. Introduction

Decomposition techniques are based on the idea of exploiting the decomposable structure present in the formulation of a given problem so that its solution can be converted into the solution of several smaller sub-problems. One is the slave problem (SP), which is obtained by fixing a number of decision variables of the initial problem $P$ to a feasible value, and the second is the restricted master problem (RMP), which is expected to provide the optimal solution after the addition of a number of cuts. The cuts are deduced from the resolution of $SP$ in each iteration of the algorithm. In each iteration, cuts are appended to $RMP$, which is solved again to optimality.

Benders method (Benders, 1962) is an algorithm that has been applied successfully to a variety of applications and different fields of mathematical programming (stochastic programming, global optimization, hierarchical optimization, etc.). Benders schemas have been used for scheduling
problems such as, for example, shift scheduling problems (Rekik et al., 2008), project selection, timing and sequencing problems (Smith and Villegas, 1997) and energy management problems (Zhang and Ponnambalam, 2006). Benders method has also been applied for the resolution of scenario-based, multistage stochastic programming problems (Watkins et al., 2000) where different scenarios have to be examined. Moreover, a global optimization algorithm, in combination with Benders algorithm, has been used to address the non-convexity nature of the mixed integer non-linear problem (Zhu and Kuno, 2003). Finally, Benders algorithm has been applied in a mixed-integer bi-level optimization problem where Karush–Kuhn–Tucker optimality conditions could not be used directly to transform the bi-level problem into a single-level problem (Saharidis and Ierapetritou, 2008a).

Over the years, various techniques and strategies have been proposed to speed up the classical Benders method approach. McDaniel and Deve (1977) proposed the generation of cuts by the solution of $RMP$ relaxing the integrality constraint. The authors present some heuristic rules for determining when the integral constraint is needed for ensuring the convergence of the algorithm. The presented results appear promising although in some cases the classical algorithm can be more efficient. Cote and Laughton (1984) presented another approach for the acceleration of Benders algorithm. In the proposed algorithm, the $RMP$ is not solved to optimality but only the first integer feasible solution is used to generate the optimality or the feasibility cut by $SP$. The main disadvantage of this strategy is that by generating only the cuts associated with the solution obtained using just a feasible integer solution, the algorithm may fail to converge. The authors proposed a heuristic approach in order to choose the iterations where the $RMP$ has to be solved to exact optimality. The same authors presented a strategy using Lagrangian relaxation where subgradient optimization is used in order to modify the Lagrangian multipliers.

Rei et al. (2008) investigated how local branching can be used to accelerate Benders algorithm. By applying local branching throughout the solution process, one can simultaneously improve both the lower and the upper bounds. They also show how Benders feasibility cuts can be strengthened or replaced with local branching constraints. Zakeri et al. (1999) presented an algorithm where the cut is not computed from an optimal extreme point of the dual $SP$. When the $SP$ is very large they determine the cuts by applying a primal-dual interior-point method to the current $SP$ and terminating the solution procedure when it reaches a feasible dual solution. The authors show that these sub-optimal cuts are computationally less expensive and can produce very good results. Finally, Magnanti and Wong (1981) are the only authors who proposed a multi-generation of cuts procedure to accelerate the Benders algorithm, using what they refer to as Pareto-optimal cuts. The authors propose to add in each iteration of the algorithm the classical Benders cut and also the Pareto-optimal cut. The obtained results show a significant reduction of the convergence time of the algorithm.

The work presented in the literature has focused mainly on either reducing the number of integer relaxed master problems being solved or on accelerating the solution of the $RMP$. In this paper, we present a new strategy to produce sufficient cuts in each iteration in order to improve the efficiency of the Benders algorithm. The outline of the paper is as follows: in Section 2, we recall the classical Benders algorithm and in Section 3 the proposed strategy referred to as covering cut bundle (CCB) generation is presented. In Section 4 we present the numerical results obtained applying both the classical Benders algorithm and the proposed approach for a series of test problems in connection with two case studies. Finally, in Section 5, we present conclusions and some perspectives for further research.
2. Benders partitioning method

2.1. Introduction: variable partitioning and the master problem

We briefly recall the idea of Benders algorithm as described using the notation as in Minoux (1986); we consider the following problem assuming that it has a feasible solution:

\[ P: \begin{align*}
\Phi &= \min \ d^T x + f^T y \\
\text{subject to:} & \quad Dx + Fy \leq b \\
& \quad x \in \mathbb{R}_+^n, \quad y \in \mathbb{R}_+^m,
\end{align*} \]

where \( d \in \mathbb{R}^n, f \in \mathbb{R}^m, D \) and \( F \) are \( p \times n \) and \( p \times m \) matrices, respectively. The structure of the given problem (\( P \)) shows that the decision variables are partitioned into two sets \((x)\) and \((y)\).

Every time we have to solve \( P \) for \( y \) fixed \((y = \bar{y})\), problem \( P \) takes the following form:

\[ SP(\bar{y}): \begin{align*}
\Phi(\bar{y}) &= \min \ d^T x + f^T \bar{y} \\
\text{subject to:} & \quad Dx \leq b - F\bar{y} \\
& \quad x \geq 0.
\end{align*} \]

To be able to apply the partitioning technique suggested above, we cannot choose \( y \in Y \) arbitrarily. It is at least necessary that the problem \( SP(\bar{y}) \) has a non-empty solution set. Associated with every constraint of \( SP(\bar{y}) \) is a dual variable \( u_i \). Denoting by \( u \) the vector of dual variables, then the following lemma (Farkas, 1902; Minkowski, 1910, 1911) states that:

Problem \( SP(\bar{y}) \) has a solution \( x \geq 0 \) if and only if: \( u^T (b - F\bar{y}) \leq 0 \) for all \( u \leq 0 \) such that \( u^T D \leq 0 \) holds.

Because the cone \( U = \{u \leq 0/|u^T D| \leq 0\} \) is polyhedral, it has a finite number of generators that are denoted by \( u^1, u^2, \ldots, u^Q \) and every \( u \in U \) is a finite linear combination with non-negative coefficients of the \( u^i \) vectors \((i = 1, \ldots, Q)\). The necessary and sufficient condition of the above lemma is then equivalent to the system of inequalities \((SI)\):

\[ (SI): \begin{align*}
(u^1)^T (b - Fy) & \leq 0 \\
(u^2)^T (b - Fy) & \leq 0 \\
(u^Q)^T (b - Fy) & \leq 0.
\end{align*} \]

In practice, this system \((SI)\) contains an enormous number of inequalities, because this is the number of generators of the polyhedral cone \( U \). If \((SI)\) has no solution, this means, by construction, that there is no \( \bar{y} \in Y \) such that \( SP(\bar{y}) \) has a solution. This implies that the problem \( P \) itself has no solution that contradicts our initial assumption.

Let \( R \) be the set of vectors \( y \in Y \), that satisfy \((SI)\); then \( P \) is equivalent to the following:

\[ \min_{y \in \mathbb{R}^m} \left\{ f^T y + \min_x \{d^T x / Dx \leq b - Fy; x \geq 0\} \right\}. \]
This may be viewed as a formalization of the idea consisting in fixing \( y = \bar{y} \), then solving a subproblem \( SP(\bar{y}) \) to choose a better value of \( y \), and so on. For a given \( y = \bar{y} \in Y \), \( SP(\bar{y}) \) is referred to as the Slave Program and its dual \( SP^*(\bar{y}) \) is as follows:

\[
SP^*(\bar{y}) \quad \Omega = \begin{cases} 
\text{Max } v^T (b - F\bar{y}) \\
\text{subject to:} \\
v^T D \leq d^T \\
v \leq 0,
\end{cases}
\]

where \( v \) is the vector of dual variables associated with the constraints of \( SP(\bar{y}) \). The constraint polyhedron of \( SP^*(\bar{y}) \), \( V = \{v | v^T D \leq d^T \} \) does not depend on \( y \), and \((u^1, u^2, \ldots, u^Q)\) defined above are its extreme rays. If \( V \) is empty, then by the duality theorem either \( SP(y) \) has no solution or is unbounded. But, by definition, \( y \in Y \Rightarrow SP(y) \) has a solution; consequently, if \( V \) is empty, this is so because \( SP(y) \) is unbounded for all values \( y \in Y \). Therefore, in this case, problem \( P \) itself is unbounded. Then by agreeing to assign the value \(-\infty\) to the maximum value of \( SP^*(\bar{y}) \), if \( SP^*(\bar{y}) \) has no solution, and using the duality theorem, we can write \( P \) as

\[
\min_{y \in \mathbb{R} \cap Y} \left\{ f^T y + \max\{v^T (b - F\bar{y}) / v^T D \leq d^T \} \right\}.
\]

Assuming that \( V \) is not empty, we denote by \((v^1, v^2, \ldots, v^K)\) the vertices of the polyhedron \( V \). Problem \( P \) can then be written as follows:

\[
\min_{y \in \mathbb{R} \cap Y} \left\{ f^T y + \max_{j=1, \ldots, K} \{(v^j)^T (b - F\bar{y})\} \right\}
\]

and this problem turns out to be equivalent to the following linear program referred to as Master Program (MP):

\[
MP: \quad \Xi' = \begin{cases} 
\text{Min } z \\
\text{subject to:} \\
f^T y + (v^1)^T (b - F\bar{y}) \leq z \\
f^T y + (v^K)^T (b - F\bar{y}) \leq z \\
(u^1)^T (b - F\bar{y}) \leq 0 \\
\vdots \\
(u^Q)^T (b - F\bar{y}) \leq 0 \\
y \in Y.
\end{cases}
\]

The number of constraints of this problem is equal to the number of vertices and extreme rays of \( V \) that is usually huge. However, we know that at an optimum solution the number of saturated (active) constraints will usually be very limited, typically equal to the number of \( y \) variables. The Benders decomposition method aims at identifying the active constraints via a constraint generation procedure.

© 2009 The Authors.
Journal compilation © 2009 International Federation of Operational Research Societies
2.2. Benders method

Suppose that at some stage only a few constraints of $MP$ are known explicitly giving rise to the Restricted Master Problem ($RMP$) as follows:

$$RMP: \begin{cases} \text{Min } z \\ \text{subject to: } \\ f^T \bar{y} + (v^i)^T (b - Fy) \leq z \\ (u^i)^T (b - Fy) \leq 0 \\ \forall j \in J \subset \{1, 2, \ldots, K\}, \forall i \in I \subset \{1, 2, \ldots, Q\}, y \in Y. \end{cases}$$

Note that $J$ and $I$ are sub-sets of extreme points and extreme rays of $SP^*$ (dual $SP$). Let $(\bar{y}, \bar{z})$ be an optimal solution of $RMP$. Because $RMP$ is deduced from $MP$ by relaxing a number of constraints, $\bar{z}$ is a lower bound for $z^*$, the optimal value of $MP$ and of $P: \bar{z} \leq z^*$.

A necessary and sufficient condition for $(\bar{y}, \bar{z})$ to be an optimal solution of problem $MP$, and hence $P$, is that $(\bar{y}, \bar{z})$ satisfy all the constraints of $MP$ that are not explicitly stated in $RMP$. In order to test this condition, we shall see that it is sufficient to solve $SP^*(\bar{y})$. Now, one of the three following cases can arise:

Case 1 (feasibility cut): The optimal value of $SP^*(\bar{y})$ is unbounded. The simplex method, applied to $SP^*(\bar{y})$, then produces an extreme ray $\bar{u}$ of $U$ such that:

$$\bar{u}^T (b - F\bar{y}) > 0 \text{ and } \bar{u}^T D \leq 0.$$ 

Thus, the constraint $\bar{u}^T (b - F\bar{y}) \leq 0$ does not hold for the current solution $\bar{y}$ of $RMP$. This means that $(\bar{y}, \bar{z})$ is not a solution of $MP$. Then the constraint $\bar{u}^T (b - Fy) \leq 0$ must be added to $RMP$ to form a new augmented $RMP$.

Case 2 (termination of the algorithm): $SP^*(\bar{y})$ has a finite optimum value and we have:

$$f\bar{y} + \bar{v}(b - F\bar{y}) - \bar{z} \leq 0$$

We can then write: $f\bar{y} + \bar{v}(b - F\bar{y}) - \bar{z} \leq 0, \forall j = 1, \ldots, J$. Moreover, we have $(v^j)^T (b - Fy) \leq 0$, for all extreme rays $v^j$ of $V, (i = 1, \ldots, I)$. It follows that $(\bar{y}, \bar{z})$ is an optimal solution of $MP$, so that $\bar{y}$ is an optimal solution for $P$ and the algorithm terminates.

Case 3 (optimality cut): The optimum of $SP(\bar{y})$ is bounded and obtained at a vertex $\bar{y}$ of $V$, but in contrast to case 2, we have $f\bar{y} + \bar{v}(b - F\bar{y}) - \bar{z} > 0$. This shows that the constraint $f\bar{y} + \bar{v}(b - F\bar{y}) - \bar{z} \leq 0$ is not satisfied by the current solution $(\bar{y}, \bar{z})$ of $RMP$. The constraint $f\bar{y} + \bar{v}(b - F\bar{y}) - \bar{z} \leq 0$ must therefore be added to $RMP$ to form a new augmented $RMP$.

3. The concept of covering cuts bundle for multi-generation of cuts

3.1. Introduction

After studying the form of the cuts produced by Benders algorithm in our case studies, we observed that the cuts are most often low-density cuts. A low-density cut is a cut involving a small
number of decision variables of $RMP$; therefore, its contribution to strengthening the $RMP$ tends to be limited. The addition of a single low-density cut does not restrict significantly the solution space of the $RMP$, thus leading to increased number of iteration and resolution time. The application of a new strategy where more than one cut is produced, in each iteration of the algorithm, in order to involve more decision variables of the $RMP$, is likely to improve the overall efficiency of the algorithm. The proposed strategy uses the concept of CCB. The CCB algorithm proceeds by generating in each iteration a bundle of low-density cuts instead of a single cut. Note that in general a bundle of low-density cuts is more desirable for acceleration of the algorithm than a cut corresponding to the sum of these low-density cuts, which is a high-density cut.

To the best of our knowledge, the concept of CCB generation and its use for speeding up the Benders algorithm has not been proposed earlier. In Section 4.1, we present two applications where the cuts produced by the classical Benders algorithm are low-density cuts and this leads, under the standard implementation, to a slow convergence of the algorithm. The CCB algorithm can be viewed as a generalization of the method proposed in Gabrel et al. (1999) and Minoux (2001) for the special case of min cost multi-commodity flow problems. In the present paper, we investigate a generalization of this strategy to any problem that is amenable to the classical Benders method. This strategy implies the generation of a bundle of cuts in each iteration by an auxiliary problem using the values obtained by the resolution of the last $RMP$. The produced bundle of cuts is intended to involve most decision variables of $RMP$, keeping the desirable form of low-density cuts. Thus, it is expected that the domains of most $RMP$ decision variables will be restricted. Different multi-generation strategies most often restrict the values of certain decision variables and leave the others unrestricted. In contrast, the CCB strategy, although it results in larger $RMP$, results in stronger restriction of the solution space, leading to a faster convergence of the Benders algorithm (see Tables 1 and 2).

The idea of CCB generation can be explained as follows: assuming that problem $SP(\bar{y})$ presented in Section 2.1 is infeasible, then a feasibility cut is deduced from the optimal dual solution of the following $SP$:

$$SP(\bar{y}): \begin{cases} \Psi = \text{Max} - \xi \\ \text{subject to:} \\ DX - \xi e \leq b - F\bar{y} \\ x, \xi \geq 0 \end{cases}$$

Its dual has the following form:

$$SP^*(\bar{y}): \begin{cases} \Psi' = \text{Min} u^T(b - F\bar{y}) \\ \text{subject to:} \\ u^T D \geq 0 \\ u^T e = -1 \\ u \leq 0. \end{cases}$$
Whenever an optimal $u$ is obtained such that $u^T(b - Fy) > 0$, the following feasibility cut results:

$$
u^T(b - Fy) \leq 0 \Rightarrow u^TFy \geq u^Tb.$$  

In CCB, the objective is to examine in each iteration the cut produced by the classical Benders algorithm and find which variables are not “covered” by this cut (the exact meaning of the “covered” term is made precise in Definition 1 below). Next we generate a second cut “covering” at least one of those variables, update the set of variables not covered yet and continue this procedure until all decision variables of $RMP$ are covered or a predefined maximum number of covered decision variables is reached.

Definition 1: A variable $y_j$ is said to be $\alpha$-covered in a cut of the form $\sum_j (u^TF_j)y_j \geq u^Tb$ if $|(u^TF_j)_j| \geq \alpha \max\{|(u^TF_j)_j|\}$ where $\alpha$ is a given parameter chosen in $[0, 1]$.

Definition 2: We call $\alpha$-CCB, a set of cuts such that each variable $y_j$ is $\alpha$-covered in some cut of the bundle.

Remark: The above definitions are used in the case of a feasibility cut. In the case of an optimality cut, the vector $v \in V$ is used instead of the vector $u \in U$.

3.2. Generation of cuts

At each iteration of the classical Benders algorithm, given the current optimal solution $(\hat{y})$ of the $RMP$, we solve the corresponding $SP$ and we produce one of the following cuts:

- $u^TFy \geq u^Tb$ or
- $v^TFy - f^Ty \geq -z + v^Tb$

where the extreme ray $u$ or the extreme point $v$ is the current optimal dual solution. The coefficient of $y_j$ in the produced cut is equal to $(u^TF)_j$ or $(v^TF)_j$. In the two applications presented in the following section, on average, 80% of these coefficients are equal to zero, which confirms that the cuts tend to have a low density. The basic idea studied here is the generation of an $\alpha$-CCB where the maximum number of variables of $RMP$ is considered. In order to generate the $\alpha$-CCB, we will consider bounds inducing constraints on $u$ or $v$ and add them to $SP^*$. These constraints have the following form:

$$LB_j \leq (u^TF)_j \leq UB_j \quad \text{or} \quad LB_j \leq (v^TF)_j \leq UB_j,$$

where the parameter $LB_j$ is the lower bound on the coefficient of the variable decision $y_j$ and $UB_j$ is the upper bound on this coefficient. Including these additional constraints in $SP^*(dual \ SP)$
and assuming that the current \( SP \) is infeasible lead to the following Auxiliary Dual Problem (\( ADP \)):

\[
\text{Min } u^T (b - F\bar{y}) \\
\text{subject to:} \\
\quad u^T D \geq 0 \\
\quad (-u^T F)_j \geq -UB_j \forall j \\
\quad (u^T F)_j \geq LB_j \forall j \\
\quad u \leq 0.
\]

As will be shown below, this problem will be solved for different values of \( LB_j \) and \( UB_j \) in order to cover the decision variable \( y_j \) of \( RMP \). Introducing two new sets of variables \( \theta_j \) and \( \mu_j \), for the additional inequalities, the corresponding auxiliary primal problem (\( APP \)) takes the following form:

\[
\text{Max } -\xi - \sum_j UB_j \theta_j + \sum_j LB_j \mu_j \\
\text{subject to:} \\
\quad Dx - \sum_j F^j \theta_j + \sum_j F^j \mu_j - \xi e \leq b - F\bar{y} \\
\quad x, \theta, \mu, \xi \geq 0.
\]

Notice that the \( APP \) problem has the same complexity as the current \( SP \) because it has the same number of constraints relaxed by the auxiliary variables (\( \theta_j \) and \( \mu_j \)), and similar to the \( SP \) problem, is decomposable or easily solvable. Thus, the solution of \( APP \) does not have a negative effect on the total CPU time. This statement is based on the idea that the CCB approach is appropriate for the case where the \( RMP \) is a more difficult problem than the \( SP \) and \( APP \). In order to generate a cut where a decision variable \( y_{j_0} \) of \( RMP \) is considered \( \alpha \)-covered in the cut, we set up the current \( APP \) as follows: we update the right-hand side of \( APP \) using the current optimal solution \( \bar{y} \) of \( RMP \) and we fix for \( j = j_0 \) the coefficient of \( \theta_{j_0} \) and \( \mu_{j_0} \) in the objective function to be: \( LB_{j_0} = UB_{j_0} = +\eta \) or \( \eta \). After we solve the \( APP \), we generate a cut that has exactly the same form as the feasibility Benders cut using the optimal value of dual decision variables. Notice that in the case where the current \( SP(\bar{y}) \) is feasible a cut with exactly the same form as the Benders optimality cut is generated. The parameter \( \eta \) is the average of the coefficients of \( \alpha \)-covered decision variables in the classical Benders cut and takes on the following value in each iteration:

\[
\eta = \frac{1}{k} \sum_{j \in \{j : |(u^T F)_j| \geq \alpha \max_{j} |(u^T F)_j| \}} |(u^T F)_j|.
\]

Notice that the \( u \) vector is the optimal solution of \( SP^n \) and \( k \) represents the total number of decision variables that are \( \alpha \)-covered in the classical Benders cut \( (k = \sum_j, \forall j : |(u^T F)_j| \geq \alpha \max_{j} |(u^T F)_j|) \). Fixing the coefficients \( LB_{j_0} \), \( UB_{j_0} \) of \( APP \)'s objective function to a non-zero value, we are ensuring that at least the variable decision \( y_{j_0} \) will be \( \alpha \)-covered in the next cut generated by the CCB procedure. The variable \( y_{j_0} \) will be \( \alpha \)-covered because these coefficients correspond to the upper and lower bound of a constraint in \( ADP \), which bounds the coefficient of the corresponded variable \( y_{j_0} \) in the produced cut that will be generated. Note that this procedure
does not constrain the values of the other coefficients. Notice that in the general form of the CCB algorithm the values of \( LB_j \) and \( UB_j \) in \( APP \) are

- \( LB_j = -\frac{\eta}{a}, UB_j = \frac{\eta}{a} \) for \( j \neq j_0 \)
- \( LB_{j_0} = UB_{j_0} = +\eta \) or \(-\eta\).

The sign of the parameter \( \eta \) depends on the value of the dual variable of \( SP(\vec{y}) \) and the corresponding element of matrix \( F \) (\( F^j \)):

- if \( v_j F^i (or u_j F^i) \geq 0 \) then \( LB_j = UB_j = +\eta \) and if \( v_j F^i (or u_j F^i) \leq 0 \) then \( LB_j = UB_j = -\eta \) for \( j = j_0 \),
- if \( v_j F^i (or u_j F^i) \geq 0 \) then \( LB_j = 0 \) and \( UB_j = \frac{\eta}{a} \) and if \( v_j F^i (or u_j F^i) \leq 0 \) then \( LB_j = \frac{-\eta}{a} \) and \( UB_j = 0 \) for \( j \neq j_0 \).

Finally, the choice of the \( y_{j_0} \) decision variable to be covered in the next cut is based on the corresponding element of matrix \( F \). If \( F_{j_0} \) is not equal to zero, then the corresponding \( y_{j_0} \) could be the next candidate decision variable.

3.3. Benders with multi generation of cuts

In the previous subsection, the procedure of the production of a cut, guaranteeing that a specified decision variable is \( z \)-covered, is presented. This procedure is important in order to restrict the solution space of the \( RMP \) in a significant way but it is not enough. Our idea is to produce not only one extra cut by \( APP \) but a number of cuts (\( z \)-CCB) where we guarantee that other variables are \( z \)-covered. In the CCB strategy not only the \( SP \) is solved but we also have the successive resolution of \( APP \) using the same optimal solution of the current \( RMP \). In each resolution of \( APP \), the parameters \( LB_j, UB_j \) are changed and fixed to a certain value for the generation of a new cut. The complete procedure implementing CCB is as shown in the schema in Fig. 1.

In the general form of the CCB algorithm, we solve problem \( APP \) for different values of \( LB_j \) and \( UB_j \). The form \( APP \) is preferable to the form \( ADP \) because only the objective function is changing during the application of the procedure of CCB. In this way, in order to produce an additional cut, we can use the last optimal basis as a starting basis for the new \( APP \) instead of solving it from scratch. Before resolving the \( APP \), the cut produced is added to \( RMP \) and for another not yet \( z \)-covered variable \( y_{j_0} \), the parameters \( LB_{j_0}, UB_{j_0} \) are fixed equal to \( \eta \) (or \(-\eta\)). At the same time the bounds \( LB_{j_0}, UB_{j_0} \) of the coefficient of the variable \( y_{j_0} \) are re-initialized. A second cut is produced and the iterations continue. The procedure stops when a predetermined maximum number of cuts has been attained or when all possible decision variables of the \( RMP \) have been \( z \)-covered. Notice that the cuts that are considered are the ones that are not satisfied by the current solution of the \( RMP \). If a cut is satisfied by the current solution then it is not added to the \( RMP \) even if it is part of the bundle of cuts in order to keep \( RMP \) as small as possible. Before the production of this \( z \)-CCB, the classical Benders procedure is applied to produce a classical cut from the optimal solution of \( SP \). After the generation of this cut, a test is performed in order to determine which variable has been \( z \)-covered in order to produce the \( z \)-CCB that covers other decision variables.
In the computational results to be presented in Section 4, the CCB strategy has been implemented as follows: each time we generate a new cut, we record the index $j_0$ of the last $\alpha$-covered decision variable ($y_{j_0}$). In the next cut generation step we cover the next decision variable after $y_{j_0}$ that is not $\alpha$-covered. This procedure ensures that all possible decision variables would be $\alpha$-covered at the end of the current iteration. Notice that when the last variable has been $\alpha$-covered then the algorithm returns to the beginning of the list of variables. Our numerical results, presented in the following section, show that the number of variables that are $\alpha$-covered, in this type of cuts, is at least as many as the ones in the cut produced by $SP$.
4. Numerical examples

4.1. Case studies

We have applied the Benders method and the CCB generation technique to problem instances arising from two different applications where Benders algorithm seems to be a good decomposition technique to be applied but its convergence is slow. These applications were selected because they correspond to the situation where the standard implementation of Benders algorithm tends to generate low-density cuts, resulting in a high CPU resolution time. After the application of CCB to the classical Benders algorithm for both examples for a majority of instances a sufficient reduction regarding both the number of iterations and the resolution time of the algorithm is observed.

Benders method is usually used when the initially developed model has complicating decision variables (Conejo et al., 2006). Decomposing the problem using these variables resulted in a series of sub-problems that are easier to solve (e.g. continuous problems). Complicating decision variables could be variables that make the problem non-convex as for example integer decision variables and/or a group of decision variables that appear in all or in most of the constraints. Herein, for both case studies, the developed models are linear mixed integer where the integer part corresponds to a series of binary variables that appear in many constraints. Summarizing, the CCB strategy is an appropriate strategy for the cases where the SP is not a complicated problem, RMP is difficult to solve and the classical implementation of Benders algorithm yields low-density cuts. These are all characteristics of the case studies studied here.

The first series of test problems that actually motivated the new approach presented here arose from Saharidis (2006). In this case study, the problem addressed concerns the development of a general model for scheduling of crude oil in a refinery. The model provides the optimal plan for the loading and unloading of crude oil to tanks by ships and/or pipelines and to crude distillation units (CDU) and/or another system of pipelines. The system is complicated due to the binary variables that have to be introduced to represent the different mixing strategies and different distillation options. The developed mixed integer linear program is decomposed on two models: the first one is the RMP and defines what is the schedule of loading and unloading of the available tanks and the second one is the SP, which, based on this information, defines the flow between the tank and upstream system (ports and pipelines) and between the tanks and the downstream system (CDU and other pipelines).

In order to further assess the efficiency of the CCB strategy, we also considered a second case study. This also concerns a scheduling problem for multi-product, multi-purpose batch plants. A detailed description of the underlying problem can be found in Ierapetritou and Floudas (1998). In brief, the problem considers the optimization of the plant capacity to improve a given economic objective. Usually profit, cost or production makespan is considered. This model in general describes scheduling problems in a variety of industrial sectors including pharmaceutical and chemicals. A continuous event-based time representation is used, resulting in a model with binary and continuous decision variables. The binary variables represent the assignment of tasks to units in a certain event point and the continuous variables represent the amount of material undertaking a task in a unit, the amounts of final products to deliver to the market, the starting and finishing time of the tasks in a unit and finally the total makespan. The decomposition of the initial problem is applied by partitioning the binary and continuous decision variables that represent the amount...
of material undertaking a task in a unit. Decomposing not only the binary variables but also these specific continuous variables, the system is decomposed into two separate blocks (two SP). The first one satisfies the material balance and capacity constraints whereas the second one satisfies the timing constraints due to task duration and sequence requirements in the same or different production units. Finally, the RMP is the problem that defines in each unit which task is going to be assigned and which will be the amount of material undertaking each task.

4.2. Results

In order to accelerate Benders algorithm the priority is to reduce the total number of RMP problems that need to be solved. Decreasing the number of main iterations usually results in a reduction of the resolution time. Resolution time is the time that the algorithm spends in order to solve the RMP but also a series of APP problems. Reducing the number of iterations results in a significant reduction due to a smaller number of RMPs that have to be solved. However, more time is required to build the additional cuts. Thus, in order to reduce the total resolution time, the gain achieved by the reduction of iterations must be greater than the time spent in order to produce the additional cuts or we have to apply parallel optimization strategies as we suggest at the end of the paper. In Table 1, we present some examples that correspond to case study 1 (Saharidis, 2006; Saharidis and Ierapetritou, 2008; Saharidis et al., 2009) where we show that the use of CCB indeed significantly improves the efficiency of Benders algorithm. A reduction of up to 98% for some examples is observed concerning the resolution time and the total number of iterations. In Table 2 we present another series of numerical examples that correspond to case study 2 (Ierapetritou and Floudas, 1998) in order to further assess the efficiency of CCB generation. All results presented in this paper have been obtained on Pentium (R) 4, CPU 2.40 GHz, RAM 1 GB and CPLEX 9.0 using a C++ implementation of the proposed approach, and have been obtained using $\alpha = 0.1$.

In Tables 1 and 2, we display the total number of iterations needed until exact optimality is reached for original Benders and CCB procedures, as well as the total running time (in seconds). The relative difference between the two approaches is given in columns 8 and 9 of each table. Finally, in the last two columns, we show the average density of the cuts produced by the original Benders and the total number of cuts produced in each iteration of the CCB generation. We notice that the cuts generated in CCB generation have a density similar to the cuts produced by the original Benders.

As illustrated in Table 1, for the first series of experiments using CCB generation, much better results were obtained than with the original Benders decomposition. With CCB generation, for the 15 examples tested an optimal solution could be obtained, requiring a significantly reduced CPU time and number of iterations compared with the standard implementation of Benders method. The difference between the two approaches is higher for the problems that require a long resolution time using the classical Benders algorithm. For example, if we look at example 1, 3 and 5 where we generate up to 100 cuts per iteration, the reduction of the CPU time is >90%. It should also be noticed that the decrease in the CPU time and number of iterations is higher when the density of the cuts is low. However, as shown in examples 14 and 15, we still observe a significant decrease in CPU time and iteration number even when the density of the cuts is higher (22–18%).

As shown in Table 2, the average density of cuts produced in the second series of experiments is higher than in the previous case. This leads to a smaller decrease in the CPU time and total
number of iterations as compared with case study 1. Note that examples 16–30 correspond to the problem presented in Ierapetritou and Floudas (1998) for different time horizons and different numbers of event points. In general, the decrease in the CPU time is higher (on average 45%) than the decrease in the number of iterations (on average 38%). For this case study, the CCB generation has the same behavior as in case study 1. When the density of the cuts is low (cf. ex. 27) the relative difference between CCB generation and classical Benders becomes higher (up to 98% for CPU time and up to 91% for iteration number). We notice that the total number of binary variables in this case study is smaller than in case study 1. The number of cuts generated in each iteration of CCB generation depends on the difference between the number of binary variables

<table>
<thead>
<tr>
<th>Example</th>
<th>Number of Benders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Var. Int./Cont. (Total)</td>
</tr>
<tr>
<td></td>
<td>Time</td>
</tr>
<tr>
<td>Ex.1</td>
<td>720/1200 (1920)</td>
</tr>
<tr>
<td>Ex.2</td>
<td>288/480 (768)</td>
</tr>
<tr>
<td>Ex.3</td>
<td>360/600 (960)</td>
</tr>
<tr>
<td>Ex.4</td>
<td>480/840 (1320)</td>
</tr>
<tr>
<td>Ex.5</td>
<td>504/864 (1368)</td>
</tr>
<tr>
<td>Ex.6</td>
<td>420/720 (1140)</td>
</tr>
<tr>
<td>Ex.7</td>
<td>480/840 (1320)</td>
</tr>
<tr>
<td>Ex.8</td>
<td>420/720 (1140)</td>
</tr>
<tr>
<td>Ex.9</td>
<td>1344/768 (2112)</td>
</tr>
<tr>
<td>Ex.10</td>
<td>504/864 (1368)</td>
</tr>
<tr>
<td>Ex.11</td>
<td>588/1008 (1596)</td>
</tr>
<tr>
<td>Ex.12</td>
<td>672/1152 (1824)</td>
</tr>
<tr>
<td>Ex.13</td>
<td>576/960 (1536)</td>
</tr>
<tr>
<td>Ex.14</td>
<td>504/864 (1152)</td>
</tr>
<tr>
<td>Ex.15</td>
<td>540/960 (1500)</td>
</tr>
</tbody>
</table>

Ex., example; iter., iteration; var., variable; con., constraint; int., integer, cont., continuous.
that are not \(z\)-covered by a classic Benders cut and the total number of binary variables. Hence, in case study 2, the number of cuts in the \(z\)-covering bundle is lower than in case study 1.

In almost all the results obtained, the application of the CCB to Benders algorithm results in a significant reduction in terms of CPU time and total number of iterations. For the example presented in Table 3, which corresponds to example 1 in Table 1, the classical Benders algorithm finds the optimal solution in 54,010 s CPU time after 2297 iterations. Applying the CCB and generating 100 additional cuts in each iteration, a reduction of 98% of the resolution time and 98% of the number of iterations is observed (cf. Table 3). In the same example there is a minimum reduction of 52.5% of the total number of iterations and 40% of CPU time. We have to pay

<table>
<thead>
<tr>
<th>Example</th>
<th>Number of Variable Int./</th>
<th>Classical Benders</th>
<th>CCB</th>
<th>Relative difference</th>
<th>Average density of Benders cuts (%)</th>
<th>Maximum number of cuts generated per iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Var. Int./ (Cont. (Total))</td>
<td>Time</td>
<td>Iter.</td>
<td>Time</td>
<td>Iter.</td>
<td>Time (%)</td>
</tr>
<tr>
<td>Ex.16</td>
<td>100/390 (490)</td>
<td>5549</td>
<td>69 s</td>
<td>34</td>
<td>49 s</td>
<td>24</td>
</tr>
<tr>
<td>Ex.17</td>
<td>120/468 (588)</td>
<td>6657</td>
<td>7022 s</td>
<td>1014</td>
<td>3272 s</td>
<td>799</td>
</tr>
<tr>
<td>Ex.18</td>
<td>120/468 (588)</td>
<td>6657</td>
<td>960 s</td>
<td>566</td>
<td>529 s</td>
<td>342</td>
</tr>
<tr>
<td>Ex.19</td>
<td>120/468 (588)</td>
<td>6657</td>
<td>72 s</td>
<td>216</td>
<td>60 s</td>
<td>152</td>
</tr>
<tr>
<td>Ex.20</td>
<td>140/546 (686)</td>
<td>7765</td>
<td>26,111 s</td>
<td>1012</td>
<td>11,401 s</td>
<td>601</td>
</tr>
<tr>
<td>Ex.21</td>
<td>140/546 (686)</td>
<td>7765</td>
<td>36,663 s</td>
<td>2727</td>
<td>22,513 s</td>
<td>2014</td>
</tr>
<tr>
<td>Ex.22</td>
<td>140/546 (686)</td>
<td>7765</td>
<td>2285 s</td>
<td>624</td>
<td>1391 s</td>
<td>489</td>
</tr>
<tr>
<td>Ex.23</td>
<td>140/546 (686)</td>
<td>7765</td>
<td>137 s</td>
<td>260</td>
<td>111 s</td>
<td>96</td>
</tr>
<tr>
<td>Ex.24</td>
<td>140/546 (686)</td>
<td>7765</td>
<td>72 s</td>
<td>16</td>
<td>27 s</td>
<td>11</td>
</tr>
<tr>
<td>Ex.25</td>
<td>160/624 (784)</td>
<td>8873</td>
<td>44,042 s</td>
<td>2915</td>
<td>26,989 s</td>
<td>2002</td>
</tr>
<tr>
<td>Ex.26</td>
<td>160/624 (784)</td>
<td>8873</td>
<td>63,068 s</td>
<td>3612</td>
<td>36,514 s</td>
<td>2292</td>
</tr>
<tr>
<td>Ex.27</td>
<td>160/624 (784)</td>
<td>8873</td>
<td>3785 s</td>
<td>615</td>
<td>167 s</td>
<td>51</td>
</tr>
<tr>
<td>Ex.28</td>
<td>160/624 (784)</td>
<td>8873</td>
<td>69 s</td>
<td>85</td>
<td>31 s</td>
<td>26</td>
</tr>
<tr>
<td>Ex.29</td>
<td>180/702 (882)</td>
<td>9981</td>
<td>29,642 s</td>
<td>1915</td>
<td>11,714 s</td>
<td>978</td>
</tr>
<tr>
<td>Ex.30</td>
<td>180/702 (882)</td>
<td>9981</td>
<td>12,782 s</td>
<td>819</td>
<td>6624 s</td>
<td>242</td>
</tr>
</tbody>
</table>

Ex., example; iter., iteration; var., variable; con., constraint; int., integer, cont., continuous.
attention to the case of 120 additional cuts, where the convergence of the algorithm becomes slow due to the time spent in order to generate the cuts. If we choose to produce a smaller number of cuts (e.g. 80 or 100), the algorithm converges faster. We should point out that even if in some cases the resolution time is worse than in the classical Benders algorithm, the number of iterations is still smaller.

The results presented in Table 3 show that the performance of CCB is related to the choice of APP’s parameters. It is found that the choice of the number of cuts produced in each iteration influences the behavior of the algorithm. In practice, some parameter tuning will be necessary in order to obtain the maximum reduction in the number of iterations or in the overall computing time. These parameters are the value of $\alpha$ and the maximum number of cuts generated in each iteration of the algorithm.

Generally, the CCB is a good strategy in order to accelerate the convergence of Benders algorithm. The only drawback may be the time spent to solve the APP programs. To address this issue, we plan to use some ideas from the parallel optimization literature in the implementation of the CCB algorithm. The main idea is to construct the $\alpha$-covering bundle while the RMP is being solved. This will involve solving a series of APP programs to produce the additional cuts using the last optimal solution obtained by the RMP. Using this parallel procedure, the $\alpha$-covering bundle will be constructed without any increase in the resolution time because RMP, which is usually an integer program, takes much longer than the solution of the required APP problem.

We have to ensure that the CCB strategy is also applicable in all other cases where Benders cuts have a medium or a high density. In the case of medium-density cuts, the CCB may not be as effective as in the case of low-density cuts but may still decrease the CPU resolution time and the total number of iterations. On the other hand, in the case of high-density cuts, we do not suggest the CCB strategy because only a few additional cuts could be produced in each iteration.

5. Conclusions and perspectives

A new strategy for Benders method has been discussed and applied in two case studies: the scheduling of crude oil and the scheduling of batch plants. The presented examples illustrate the applicability and efficiency of the new strategy called CCB generation. All the presented
examples illustrate that CCB results in a significant decrease of the number of iterations of the Benders algorithm as well as the CPU resolution time.

As can be observed from the numerical examples, the CCB strategy is important especially when the \textit{RMP} is a difficult problem. The \textit{RMP} in the first case study is more complicated than in the second one. Adopting the CCB strategy, the number of main iterations almost always decreases, resulting in a better resolution time because the \textit{RMP} has to be solved significantly less frequently.

The acceleration of the CCB strategy can be addressed using parallel optimization because the \textit{SP}, the \textit{APP} linear programs, which yield the \(z\)-CCB, can be solved in parallel with the resolution of \textit{RMP}. The \textit{RMP} is usually an integer linear program and requires a substantial resolution time. During this time and using a feasible but not yet optimal solution, the \textit{SP} can generate a valid \(z\)-covering cut bundle that will be added to \textit{RMP} after its resolution. Finally, parallel optimization can also be used in order to exchange sub-optimal information between the \textit{SP\textsuperscript{*}}, \textit{APP} and \textit{RMP} problems.

\textbf{Acknowledgements}

M. Ierapetritou would like to gratefully acknowledge financial support from the National Science Foundation under the NSF CTS 0625515 grant and also the USEPA-funded Environmental Bioinformatics and Computational Toxicology Center under the GAD R832721-010 grant.

\textbf{References}


© 2009 The Authors.

Journal compilation © 2009 International Federation of Operational Research Societies